Supporting malleability in GekkoFS

Marc-André Vef, Alberto Miranda, Ramon Nou, André Brinkmann

Johannes Gutenberg University Mainz, Germany

Barcelona Supercomputing Center, Barcelona, Spain

2nd June 2022

HPC-IODC

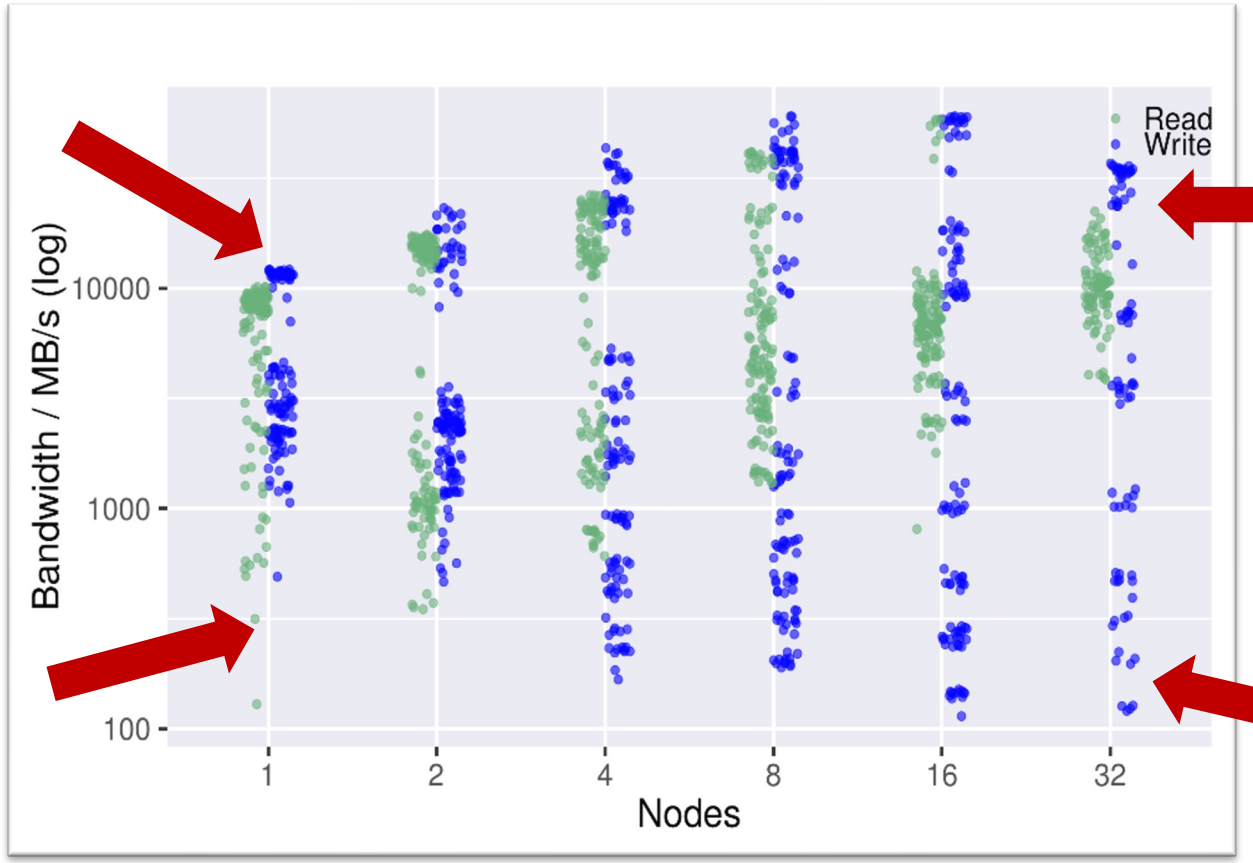
HPC I/O in the Data Center

Motivation

The cost of using the parallel file system

I/O performance varies wildly for identical workloads

Applications suffer due to PFS load!



Motivation

MareNostrum 4
Peak I/O bandwidth:
Read: 204,96 GB/s
Write: 120,89 GB/s

PFS BW per node (avg. 3456 nodes):
Read: 60,72 MB/s
Write: 35,81 MB/s

vs

Node-local Intel s3520 SSD:
Read: 450 MB/s
Write: 380 MB/s

From S. Moré, “Storage in MareNostrum 4: Petaflop System Administration” PATC 03/2019

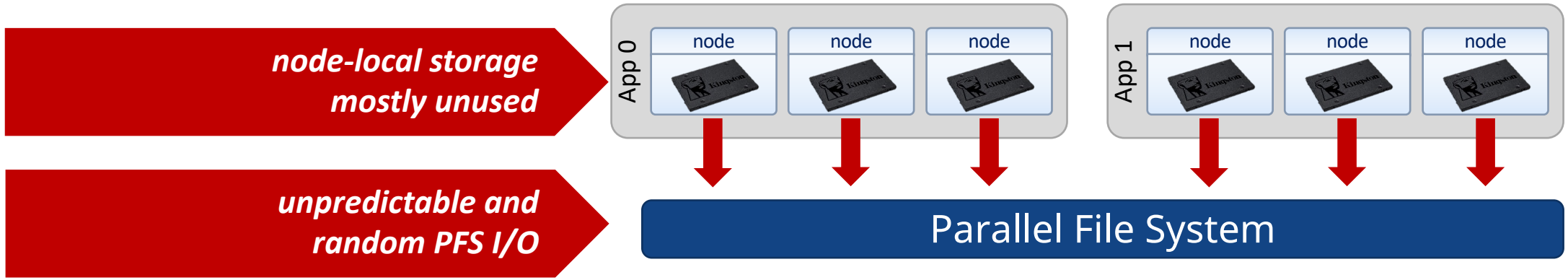
- Minimize arbitrary PFS usage: exploit the available I/O stack
- Minimize redundant data movement and schedule transfers to reduce PFS contention
- Improve data locality: Do work where data lives!

Goal

Moving from this ...

Data manipulations rely on the PFS

- Uncoordinated application I/O to/from PFS
- Node-local storage typically ignored
- Increased PFS contention and performance variability

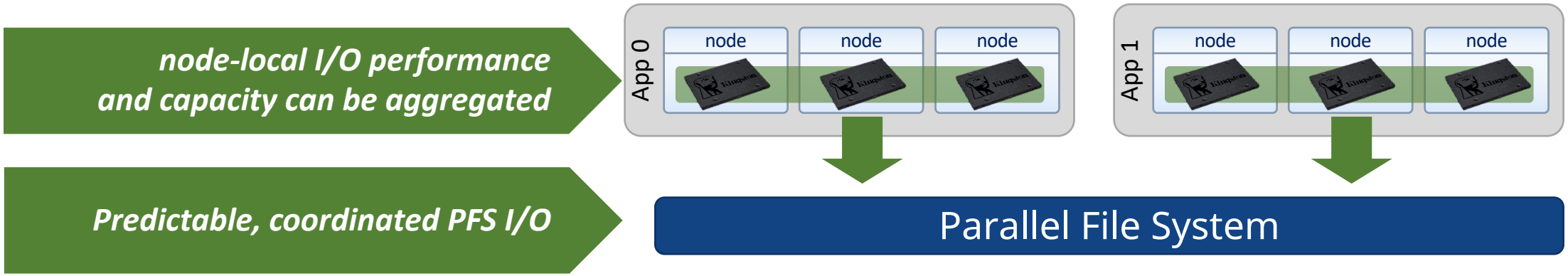


Goal

... to this

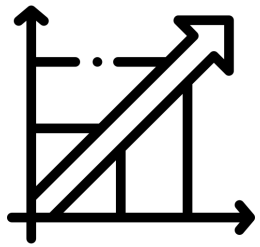
Data manipulations rely on node-local storage

- Coordinated application I/O: sequential stage-in (read) and stage-out (write) from/to PFS
- Harmful I/O patterns are absorbed by node-local storage
- Reduced PFS contention and performance variability

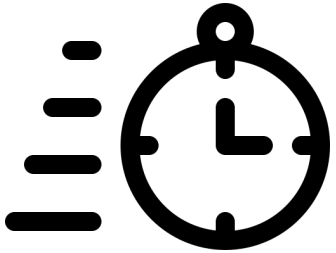



GekkoFS

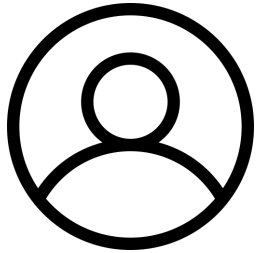
Core challenges to be addressed



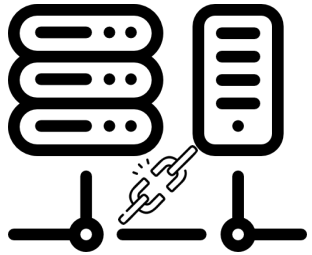
1. Scalability



2. Fast deployment

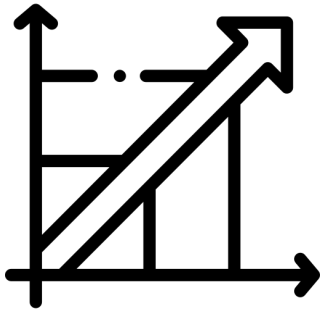


3. User space



4. Hardware independence

Core design



Let's rethink metadata handling in distributed file systems

- Directory/indirect blocks and inodes are *not* designed for parallel access
  - Leads to high code complexity, heavy communication, and intricate locking
  - Result: poor scalability (see common parallel file systems)
- Instead,
  - compute metadata destinations on the fly,
  - let the target node handle the request independently, and
  - remove most metadata (timestamps, permissions, …).

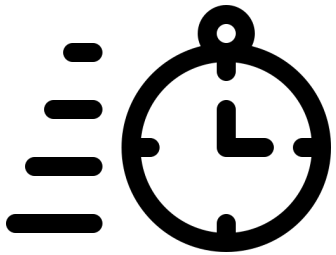
➢ **No central components are necessary**

M.-A. Vef, V. Tarasov, D. Hildebrand, A. Brinkmann.
Challenges and solutions for tracing storage systems: A case study with Spectrum Scale. In ACM Transactions on Storage, 2018

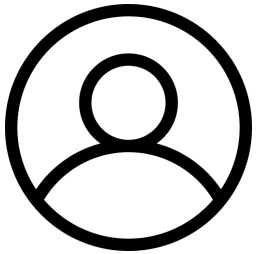
Core design

- Loosely coupled and no inter-node locking mechanisms
- Simplify file system protocols
  - No *Virtual File System* (VFS)-dictated file system protocols
- Path-based flat namespace
- Favor applications over user file system interaction
- Relax file system consistency
  - Strong consistency for direct file operations (file create/stat, write, and read)
  - Weaker consistency for indirect operations (`**ls -l**` or `**rm -rf /foo/bar/***`)

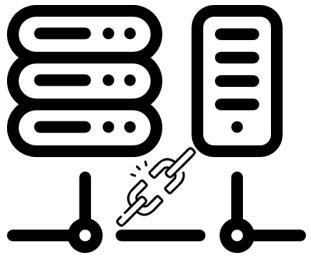
Core design



- Fast deployment: <10 seconds for 512 nodes
- Decoupled file system components



- User space file system (system call interception)
- No admin support necessary



- Hardware independence for storage and network

What GekkoFS is not

1. GekkoFS is not a long-term, general purpose PFS
   - **Ephemeral**: its lifetime is linked to an application/workflow
2. GekkoFS is not multi-user
   - Usable with **normal user privileges**
3. GekkoFS it not POSIX... mostly
   - GekkoFS **supports the POSIX I/O API** but discards some semantics in favor of performance
   - GekkoFS can also offer **specialized APIs**

What GekkoFS is

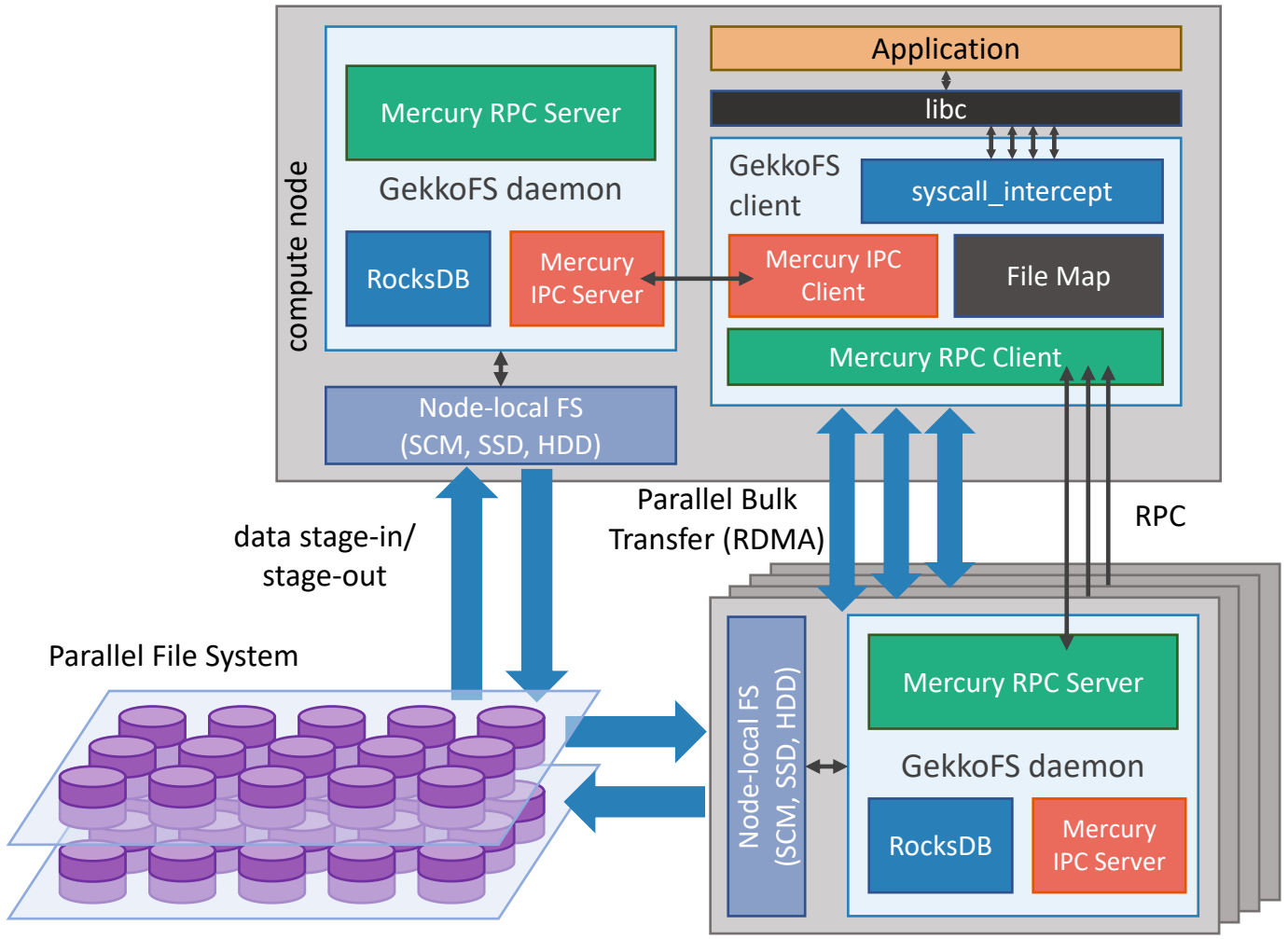
1. GekkoFS is a high-performance distributed file system for a single application
   - Allows **aggregating node-local storage** performance/capacity
   - Provides a **shared namespace** between nodes
2. GekkoFS is intended to be tuned for a specific application
   - Configurable **metadata management**: shared/non-shared, flat/hierarchical namespace, symlinks, access times updates, etc.
   - Configurable **data management**: data distribution, access consistency model, etc.
3. GekkoFS is easy to use
   - Runs in **user space** – easy installation and maintenance
4. GekkoFS is highly scalable
   - Performance of fully distributed mode **scales linearly** with the number of nodes
   - Data based on chunks: Internal **access pattern transformation**
     - Shared file vs. file per process
     - Sequential vs. random

GekkoFS architecture

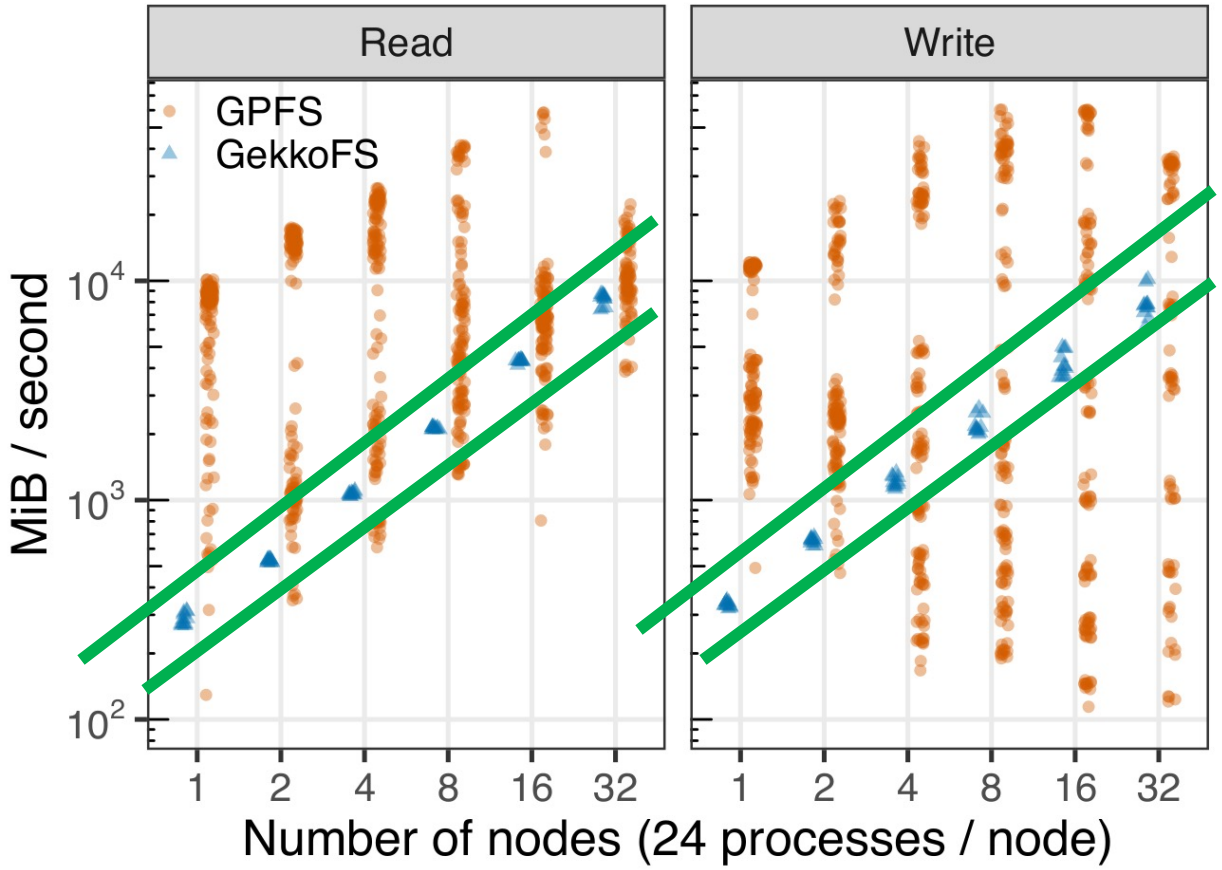
Mercury
A high-performance RPC framework from ANL
https://mercury-hpc.github.io

RocksDB
A persistent key-value store for fast storage from Facebook
http://rocksdb.org

syscall_intercept
A system call interception library from Intel
https://github.com/pmem/syscall_intercept



Performance variability revisited (MN4)

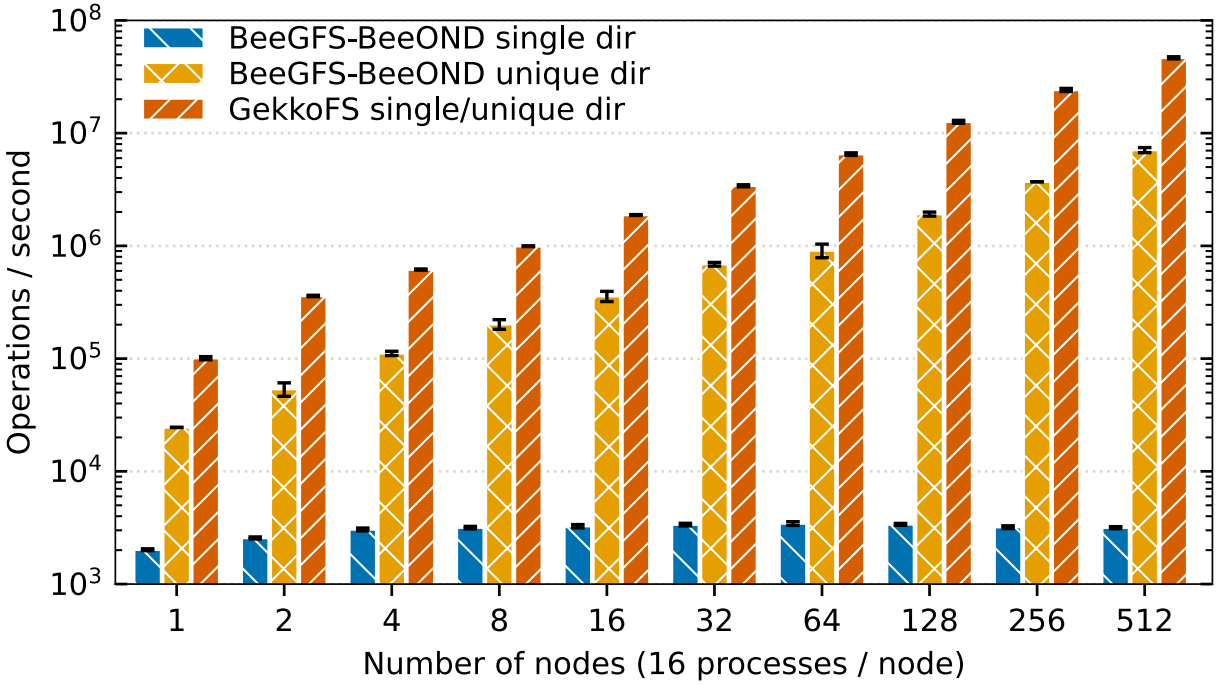


I/O performance variability is greatly reduced

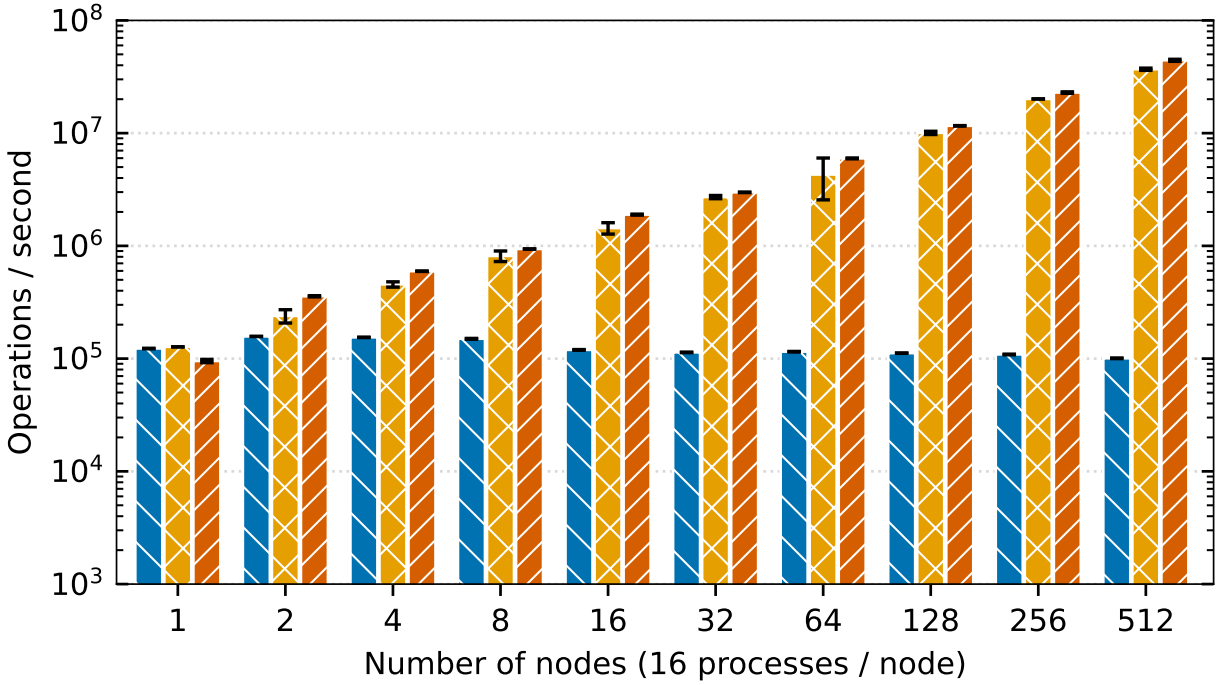
Metadata performance

GekkoFS vs. BeeGFS @ MOGON II

- GekkoFS and BeeGFS weakly scaled (100K files per process)
  - More than 819 million files in total at 512 nodes



File create performance



File stat performance

Other results

IO500

- Ranked 4th in the 10-node @ SC19
- MadFS design based on GekkoFS

Distributed deep learning with Tensorflow (Schimmelpfennig et al. @ CLUSTER '21)

- Similar step time performance as local storage
- No biases with GekkoFS

S3D

- I/O is done through PnetCDF
- 729 MPI processes; WRITE-only workload => 3476.14 GiB
- Bandwidth: 795.67 MiB/s vs 8651.79 MiB/s (+10x)

HACCIO

- Checkpoint – restart workload
- WRITE Bandwidth: 932.691 MB/s vs 946.617 MB/s (+1.7x)
- READ Bandwidth: 2458.82 MB/s vs 4208.82 MB/s (+2x)

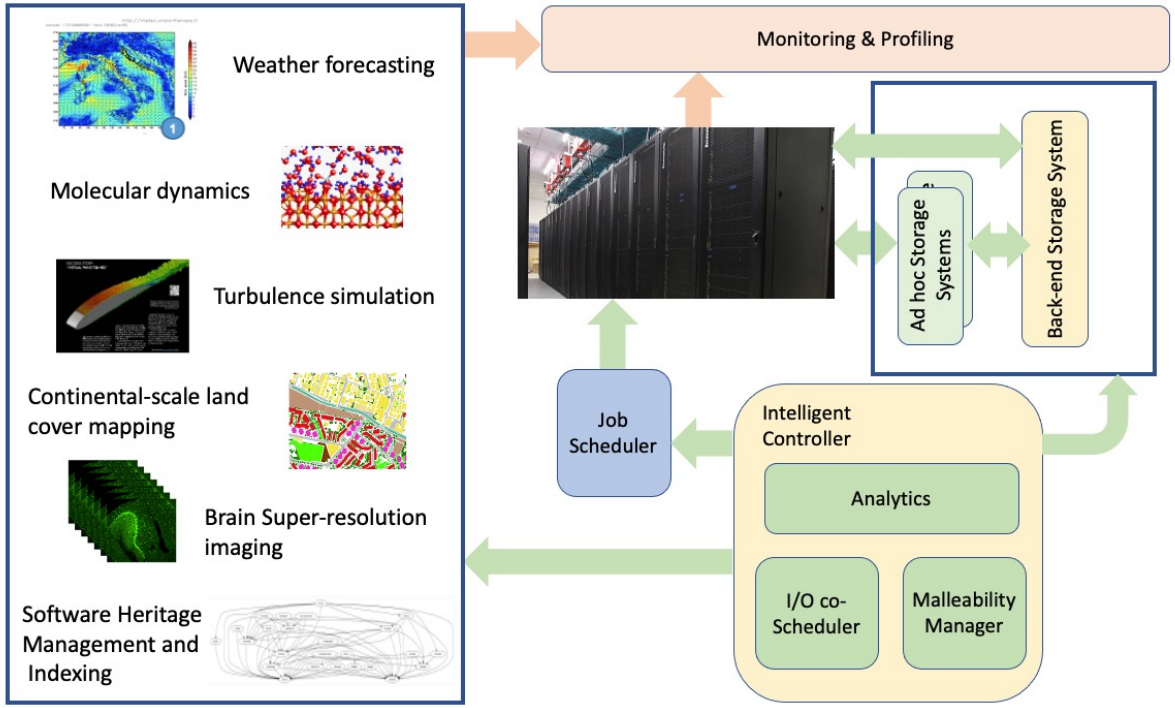
How does it work

- Clone GekkoFS
  - `git clone --recurse-submodules https://storage.bsc.es/gitlab/hpc/gekkofs.git`
- Download and compile dependencies (Spack support WIP)
  - Set `LD_LIBRARY_PATH` for dependencies
  - Download: `gekkofs/scripts/dl_dep.sh /home/foo/gkfs_deps/git`
  - Compile: `gekkofs/scripts/compile_dep.sh /home/foo/gkfs_deps/git /home/foo/gkfs_deps/install`
- Build GekkoFS
  - `cmake -DCMAKE_PREFIX_PATH=/home/foo/gkfs_deps/install -DCMAKE_INSTALL_PREFIX=<ipath> ..`
  - `make -j install`
- Start the server(s) (deployment scripts available)
  - `<ipath>/gkfs_daemon -r <data_path> -m <gkfs_mount_path> -H <hostfile_path>`
- Set the host file on a path accessible to all clients
  - `export LIBGKFS_HOSTS_FILE=<hostfile_path>`
- Use `LD_PRELOAD` to use the GekkoFS client
  - `LD_PRELOAD=<ipath>/libgkfs_intercept.so cp ~/some_input_data <gkfs_mount_path>/some_input_data`

malleable data solutions for HPC

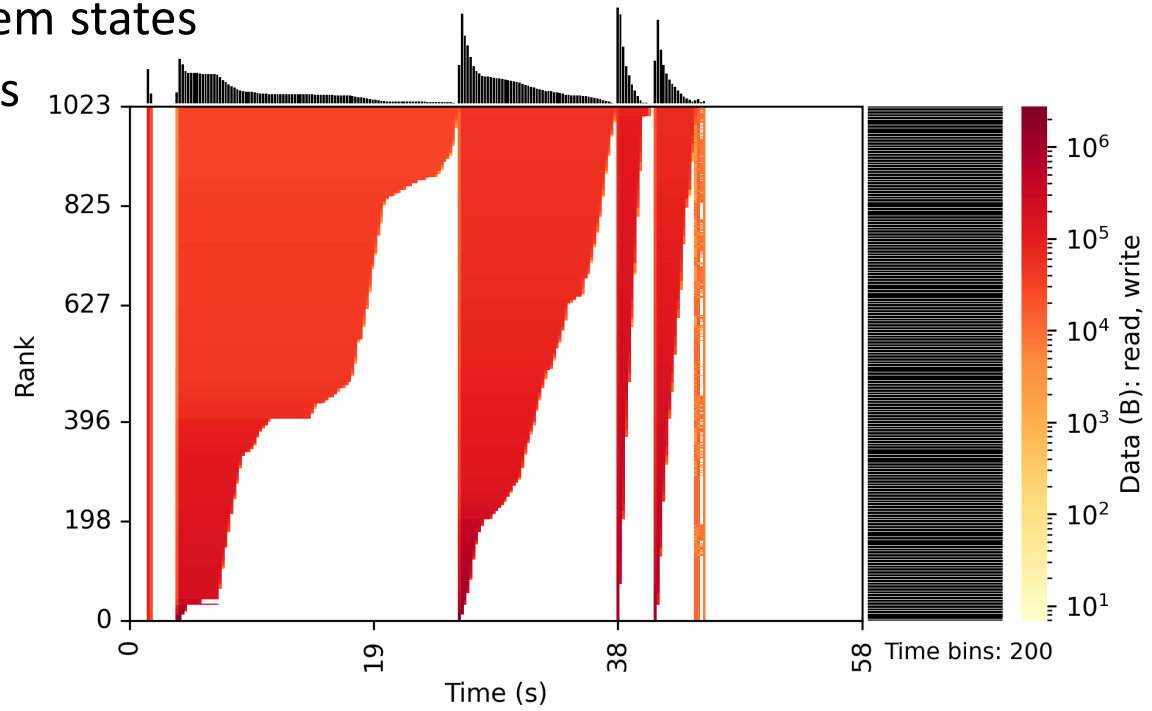
The ADMIRE project

- Adaptive multi-tier intelligent data manager for Exascale
- Develop an active I/O stack
- Dynamic adjustment of computation and storage requirements (global coordination)
- Computation and I/O malleability
- Quality-of-Service (QoS)
- Direct collaboration with HPC applications
  - Turbulence, molecule, environmental simulations
  - Life sciences
  - Deep learning
  - Software heritage
- Visit https://www.admire-eurohpc.eu



I/O malleability

- Or, the ability to adapt the storage system to a use case during runtime
- PFSs only allow limited malleability, e.g., QoS
- Requirements for I/O malleability
  - A monitoring body observing application and system states
  - A decision body when to apply malleable decisions
  - A malleable storage system
  - Available storage, e.g., node-local
- Possible malleable options
  - Extending and decreasing I/O nodes
  - Relaxing file system semantics
  - QoS, data distributions and more



Heatmap /w Darshan for Nek5000

What GekkoFS will be in ADMIRE

1. GekkoFS will support long-living workflows
   - Usable by **several applications within workflows**
   - Add **error correction** mechanisms
2. GekkoFS will be malleable during runtime
   - Increase/decrease **number of FS nodes**
   - **Control QoS**, i.e., FS bandwidths
   - Varying **caching** aggressiveness, changing FS configurations, data distributions etc.
3. GekkoFS will support fast storage technologies (e.g., persistent memory)
4. GekkoFS will integrate into the ADMIRE ecosystem
   - Connection to **I/O scheduler** controlling GekkoFS startup/shutdown and staging
   - Connection to **intelligent controller** taking system decisions
   - Connection to **monitoring module**

Thank You

JGU

- Marc-André Vef — vef@uni-mainz.de
- André Brinkmann — brinkman@uni-mainz.de

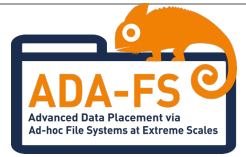
BSC

- Ramon Nou — ramon.nou@bsc.es
- Alberto Miranda — alberto.miranda@bsc.es

Acknowledgements: Supported by

Flaticon.com for Icons

GekkoFS history & publications

- Created within the German-funded ADA-FS project 

- BSC collaboration since early 2018 within the EU-funded NEXTGenIO project 

- Funded by the ADMIRE and FIDIUM projects since 2021
- GekkoFS-related publications:
  - Vef et al. @ CLUSTER conference 2018
  - Soysal et al. @ HPCS conference 2019
  - Vef et al. @ Journal of Computer Science and Technology 2020
  - Brinkmann et al. @ Journal of Computer Science and Technology 2020
  - Bez et al. @ Future Generation Computer Systems journal 2020
  - Bez et al. @ IPDPS conference 2021
  - Schimmelpfennig et al. @ CLUSTER conference 2021